Exponential Decay Function-Based Time-Aware Recommender System for e-Commerce Applications

Ayat Yehia Hassan[1], Dr.Etimad Fadel[2], Dr.Nadine Akkari[3]
Computer Science, King Abdul-Aziz University, Jeddah, Saudi Arabia[1, 2]
Computer Science, Jeddah International College, Jeddah, Saudi Arabia[3]

***Abstract*—Unlike traditional recommendation systems that rely only on the user's preferences, context-aware recommendation systems (CARS) consider the user's contextual information such as (time, weather, and geographical location). These data are used to create more intelligent and effective recommendation systems. Time is one of the most important and influential factors that affect users' preferences and purchasing behavior. Thus, in this paper, time-aware recommendation systems are investigated using two common methods (Bias and Decay) to incorporate the time parameter with three different recommendation algorithms known as Matrix Factorization, K-Nearest Neighbor (KNN), and Sparse Linear Method (SLIM). The performance study is based on an e-commerce database that includes basic user purchasing actions such as add to cart and buy. Results are compared in terms of precision, recall, and Mean Average Precision (MAP) parameters. Results show that Decay-MF and Decay-SLIM outperform the Bias CAMF and CA-SLIM. On the other hand, Decay-KNN reduced the accuracy of the RS compared to the context-unaware KNN.**

Keywords*—Time-aware recommender system; context-aware recommender system; matrix factorization; K-Nearest Neighbor (KNN); and Sparse Linear Method (SLIM)

I. Introduction and Background

Recommender systems (RS) are intelligent tools and techniques used to recommend items to a user based on his/her preferences [1]. In Ecommerce Applications, a Recommender system is used to predict the product that a user is most likely to purchase. Companies like Netflix and Amazon use recommender systems to help their users to identify the correct product or movies based on their history[2].

Context-aware recommender systems (CARS) produce more significant recommendations by optimizing preferences to suit the current situation and conditions of the user (e.g., location, time, weather, device, etc.) [15,17]. This method has been proven to be effective in improving the performance of the recommendation system [3]. One of the most important contextual information that has been used in recommender systems is time, especially in the context of e-commerce applications. The winning team of the Netflix Prize competition [5] found that using time context can significantly increase the reliability of the recommendations. As users' preferences change over time, new fashions and interests are constantly emerging [4]. For example: seasonal changes (specific holidays) leading to different shopping patterns. Also, Product popularity are in a constant change. This leads consumers to constantly change their taste. Therefore, it was necessary for the recommendation systems to consider these changes in the behavior of users. One of the factors that facilitated research on the use of time in recommendation systems is the ease of extracting it, which does not require special devices or effort.

The time Aware Recommender system (TARS) 's primary purpose is to deal with user preferences changes over time [18]. There are two types of user's feedback that can be used with the time: implicit feedback and explicit feedback. In the explicit feedback the system must ask users to provide their ratings for items directly mostly using stars. However, in the implicit feedback approach, the system automatically tracks users' preferences by monitoring the performed actions, such as which item they visited, where they clicked, which items they purchased, or how long they stayed on a web page. In the real word applications, any store can include a time-aware recommender system to work offline without the need to collect new data from users because it can simply use the user's implicit feedback (purchase data) with its timestamp. As it has been proven in this research, choosing the appropriate algorithm to use with the time data will improve the performance of the recommendation system, which will be reflected in the store's revenues. Some research discussed the impact of using time with user's explicit feedback. However, research that consider purchasing information such as [3, 4, 6] did not address the time dimension. Due to the lack of datasets that uses implicit feedback (purchase information) with time factor in the field of e-commerce, the impact of linking these two parameters in the field of e-commerce has not been studied in the literature. Thus, many of the TARS research findings don't apply to e-commerce. As this type of recommender system is concerned with not only the preferences of the user but also with the act of purchasing, which leads to increasing the store's earnings which is the main goal of any business. This work analyzes TARS with purchasing actions on an online shopping domain to achieve the goal of increasing the store revenue by enhancing the accuracy of the top 20 recommended items.

In previous research [51] we incorporated time with the Matrix Factorization (MF) algorithm to improve the recommender system accuracy. In this research, we have used the Decay function method with two other states of art algorithms to compare the results and find out the most appropriate algorithm in the field of e-commerce that can be used with the common actions (add to cart and buy). The work's key contribution is to combine time with RS using two separate methodologies (Bias and Decay) and purchasing actions for online shopping recommendations.

Bias is the first method, which uses time as the third dimension in the (user*item) rating matrix. The Decay function,

on the other hand, generates predicted ratings by combining implicit feedback (add to cart and buy) with temporal dynamics, giving the newly purchased items a higher weight than older ones. Then generating new predicted rates to replace the old rates.

In this research, we evaluated the two techniques using three cutting-edge algorithms: Matrix Factorization (MF) [7], K-Nearest Neighbor (KNN) [8], and Sparse Linear Method (SLIM) [9]. Precision, recall and Mean Average Precision (MAP) were used to evaluate the experimental results.

A. Time-Aware Recommender System Categories

Authors in [19] identified seven methods on how time factor may be used with recommender systems as follow:

- Bias: In this method, the system records the time of the user's rating. Then Time will be added to the collaborative filtering algorithm as the matrix's third dimension. Collaborative filtering will compare users, identify similar users, and then predict user ratings for unrated items.
- Decay: In this method, the system prioritizes new things above those with more recent interactions.
- Micro-profile: The system saves distinct profiles for each user at different times. For example, a user can have two profiles, one for weekends and the other for weekdays.
- Restriction: In this case, the recommender system (RS) matches the user's available time to the time when the item will be used. For example, if the user wants to eat dinner, the algorithm will only suggest restaurants that are open late.
- Time Rating: Time is used to infer user preferences using implicit feedback. For example, the longer a user spends on a product page, the more the consumer likes that product.
- Novelty: The recommendation system defines a limit date and will not recommend any items that are older than this date. For example, on a news website, RS will only recommend news that was published at least one day ago.
- Sequence: The RS tracks items that are consumed one after another or in a certain order. For example, the system will recommend more products that are typically purchased along with the consumed product.

In this paper, we investigated and compare the Bias and Decay categories by incorporating them with RSs Algorithms.

B. Recommender Systems Algorithms

There are three main techniques used in RS: content-based filtering, collaborative filtering, and hybrid filtering. However, our research scope is in Collaborative filtering algorithms (CF) which can be categorized into three main categories: neighborhood-based algorithms such as KNN, latent factor-based algorithms such as MF, and non-neighborhood or latent factor algorithms such as Sparse Linear Method (SLIM) algorithm [11-14].

- Neighborhood Based Algorithms

The K-Nearest Neighbors (KNN) algorithm depends on finding users who are similar to the current user. The behavior of the current user is thus predicted based on his/her closest neighbors [20]. The interaction matrix serves as the algorithm's initial input. The correlation between users is then used to compute similarity. The cosine similarity is applied and computed in this study as given in equation (1) [8]:

$$sim(u_a, u_b) = \frac{\sum_i r_{u_a,i} r_{u_b,i}}{\sqrt{\sum_i {r^2}_{u_a,i} \times \sum_i {r^2}_{u_b,i}}} \quad (1)$$

Where $sim(u_a, u_b)$is the correlation between user a and user b and $r_{u_a,i}$ is the rate by the user a for item i. k (number of the nearest neighbors) is a hyper-parameter that must be manually adjusted and cannot be learned by the system. The final step is to use the formula (2)[8] to calculate the rating prediction:

$$P_{u_a,i_x} = r_{u_a}^- + \frac{\sum_{n \in V}(r_{u_n,i_x} - r_{u_n}^-) sim(u_a,u_n)}{\sum_{u_n \in N} |sim(u_a,u_n)|} \quad (2)$$

N is the set of closest neighbors, while ${r^-}_{u_n}$ is the average rate that one of the users in the N set has provided. P_{u_a,i_x} denotes the predicted rate of user *a* for item *x*, r_{u_a,i_x} is the actual rate by user *a* for item *x*.

- Matrix factorization Algorithm

Matrix factorization (MF) is a form of latent factors technique used in RSs that uses Collaborative Filtering (CF). MF method begins by randomly initializing two matrices. The first is a (users*factors) matrix, while the second is a (factors*items) matrix. When these two matrices are multiplied together, we get the (users*items) matrix, which is the same size as the Rating Matrix that we're attempting to forecast [21]. The number of latent components we're utilizing to estimate the rating matrix is represented by dimension f (factors). f is usually between 10 and 250.

This method, as stated in equation (3), seeks to fill the matrices P and Q by predicting the ratings in the training set [7].

$$\min \sum_{(u_a,i_x) \in \tau} \left(r_{u_a,i_x} - P_{u_a,i_x}\right)^2 = min_{p_*,q_*} \sum_{(u_a,i_x) \in \tau} \left(r_{u_a,i_x} - q_{i_x}^T P_{u_a}\right)^2 + \lambda\left(\left\|q_{i_x}\right\|^2 + \left\|p_{u_a}\right\|^2\right) \quad (3)$$

Where $\lambda\left(\left\|q_{i_x}\right\|^2 + \left\|p_{u_a}\right\|^2\right)$ is a regularization term intended to prevent overfitting. p and q are the two factors matrices.

- Sparse Linear Method (SLIM) Algorithm

Similar to the neighborhood-based approach, the sparse linear method (SLIM) method [9] tries to reduce the error rate by using the loss function rather than similarity to determine the difference between the training set and the test set. Equation (4)[9] illustrates the error function's definition as follows:

$$\min_{w_*} \sum_{(u_a,i_x)\in\tau} \left(r_{u_a,i_x} - r_{u_a}^T w_{i_x}\right) + \lambda_1 \left\|w_{i_x}\right\|^2 + \lambda_2 \left\|w_{i_x}\right\|_1$$

$$s.t \left\|w_{i_x}\right\|_1 \geq 0 \quad (4)$$

Where *w* is the item-item similarity learned by minimizing the error function, w_{i_x} is a column from the *w* matrix, $r_{u_a}^T$ is the user rate in the training set, r_{u_a,i_x} is the predicted rate for item x, and $\lambda_1 \left\|w_{i_x}\right\|^2 + \lambda_2 \left\|w_{i_x}\right\|_1$ are regularization terms to prevent overfitting.

C. Evaluation Matrices

There are several measures for defining the quality of the Top-N Recommender system. However, the main three are precision, recall, and MAP [10].

- Precision

The percentage of recommended items in the Top-N set that are relevant is known as precision. For example, if precision at 10 in a top-10 recommended items is 90%. This indicates that 90% of the suggestions are useful to the user. According to [22], precision is calculated as follows:

$$p = \frac{\#\ relevent\ recommended\ items}{\#\ recommended\ items} \quad (5)$$

- Recall

Recall is the percentage of relevant items in the top-N recommended items. For instance, if we measured recall at 10 and discovered that it is 40%. This indicates that the top-N results contain 40% of the entire number of relevant items. The following is the definition of mathematical recall [22]:

$$r = \frac{\#\ relevent\ recommended\ items}{\#\ all\ the\ possible\ relevant\ items} \quad (6)$$

- Mean Average Precision (MAP):

MAP is the mean of average precision, of all users. MAP can be calculated as follows [22]:

$$MAP = \sum_{i=1}^{N} \frac{P@i}{N} r_i \quad (7)$$

where N is the length of the list of recommended items and P@i is the precision at item i.

The remainder of the paper is structured as follows:

Section II provides a summary of related works. Section III explains Incorporating time context into RS algorithms. The implementation and experiment are found in Section IV. Section V explains the findings and discussion. Finally, in section VI, the conclusion is presented.

II. Related Work

Many scientists investigated how time is used in RS. Some used it with rating prediction recommenders [29,27,30,33], while others used it with Top-N recommendations [25,26,37]. Precision, recall, and MAP may be applied for Top-N recommendations, whereas Mean Absolute Error (MAE) and Root mean squared error (RMSE) can be used for recommendation prediction. We will focus on Top-N recommendations in this study because they are commonly used in the commercial field, where the purpose of the recommender system is to select a few specific things that are most attractive to the user.

The recommendation process may be carried out in different ways and using a variety of algorithms. Neighborhood-based algorithms and latent factor algorithms are the most popular types of traditional recommender systems. The recommender was applied using Matrix factorization and user-based K-nearest neighbor (KNN) in [23]. They created a simulated rating utilizing the decay function to aggregate implicit feedback datasets with time dynamics. The experimental findings showed that their suggested technique is successful in the multimedia domain for rating prediction and top-N recommendation. The study's [24] objective was to enhance the performance of neighborhood-based recommender systems using the time context with the decay function in addition to ratings. The author [25] employed content-based and collaborative filtering algorithms with pre- and post-filtering that considered the time context. However, they do not provide any experimental tests on a dataset. Researchers in [25] classified the methods of using time context with RSs as decay, restriction, and novelty. Other researchers added the sequence category with the decay function [26].

the research [27] exploited using the time context to enhance the user-based collaborative filtering algorithm and proposed a weight formula to account for changes in the group users' preferences over time. Their technique raises the accuracy of the collaborative filtering (CF) algorithm on a movie dataset. A proactive Context-Aware RS is suggested by [28] for lecturers and scientists who would be providing learning resources for students. In [28], time is employed in the same way as Restriction; the RS seeks to locate learning materials that correspond to the real user's time. [29] suggested a method for predicting user preferences in a recommender system by learning the sequence of purchase history and taking preference changes into account. Using the micro profile technique, they use a Kalman filter to forecast user preference vectors from user characteristics.

Time is widely implemented in the multimedia domain [23, 24, 30-34,52] and learning domain [28, 35-37] [25, 26] according to the literature. However, in the e-commerce field, it is rarely utilized to study purchase data rather than just rating data. We can also see that user preferences fluctuate depending on the season and application domain. As a result, the method for dealing with time in one domain may not apply to another. We used the decay function to apply to the purchase date (add to cart & buying or transactions) in an online shopping system in this study.

Many studies in the literature [24,26,27,30-33,53] used the MovieLens dataset, which contains the rating from the MovieLens website (movielens.org). The ratings are connected to the timestamp. Other research [23, 29] examined the Last.fm dataset, which includes a timestamp and a user's history of listings in addition to the user's properties. The Yelp dataset in [24] only gives a comprehensive perspective of restaurants, including overall user ratings. Finally, the D-Lib Magazine dataset [26] includes articles and numerous shorter items, as

well as digital collections, calls, and notifications from its 265 issues.

According to the literature, most of the datasets used do not include the implicit feedback of purchase data (buy and add-to-cart) linked with time in the e-commerce environment. Table I shows a summary of the related work.

TABLE I. RELATED WORK

REF	Algorithm	Time-aware RS category	Implicit/explicit data	Domain
[23]	KNN MF	Decay	User View history (Implicit)	Music
[24]	KNN	Decay	Rating data (explicit)	- Movies - Business Directory Service
[25]	Hybrid (CF+ Content Based)	- Decay - Restriction - Novelty	Date, Duration, Learning Time (implicit)	Learning
[29]	Kalman filtering - Matrix factorization	Micro-profile	User View history (Implicit)	Music
[38]	CF	Other	-	-
[32]	Matrix-factorization model	Other	Rating data (explicit)	Movies
[30]	Time- and Community-Aware RS	decay	Rating data (explicit)	Movies
[31]	Temporal overlapping community detection method	Other	Rating data (explicit)	Movies
[33]	a novel dynamic recommender system.	Other	Rating data (explicit)	Movies
[36]	knowledge-driven recommender	Restriction	-	Mobile Learning on the Semantic Web
[27]	CF	Decay	Rating data (explicit)	Movie
[28]	Context-aware RS	Restriction	-	Learning
[37]	Hybrid	Sequence	-	Learning environments
[52]	Novel next-item RS	Sequence	Interval and duration (implicit)	Game-playing data
[53]	Hybrid	Other	Rating data (explicit)	Movie

In this research, we have used the Decay function method with three states of art algorithms in the field of e-commerce with two main implicit actions (add to cart and buy). The work's key contribution is to analyze the effect of combining the time context with RSs using two separate methodologies (bias, and decay) and user's implicit feedback for online shopping recommendations.

III. INCORPORATING THE EXPONENTIAL DECAY FUNCTION INTO RECOMMENDER SYSTEM

Our approach consists of three major phases as illustrated in Fig. 1. The first phase is data pre-processing where dataset converted to binary format. The second is the half-life decay function [30,31,33]. Finally, the contextual modeling where Multidimensional recommender (MD) is generated to produce the Top-N recommended items. the three phases are explained as follows:

A. Pre-Processing Phase

This phase involves converting the dataset in a compact format which includes the interaction condition and its value into a binary format that only uses 0s and 1s as shown in Fig. 2. Converting the event types (add to cart and buy) to numerical numbers is also included in this phase.

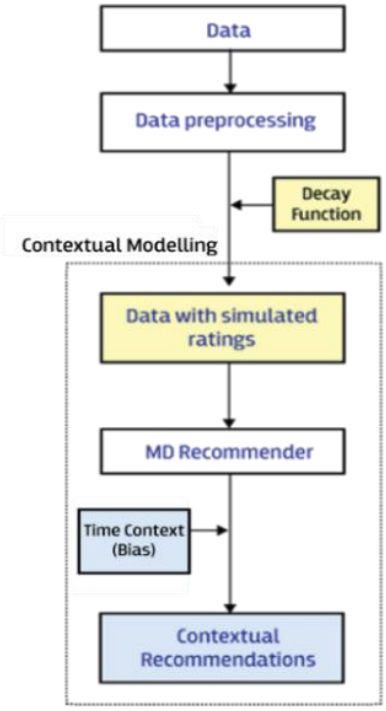

Fig. 1. Incorporating Time Dynamic with Recommender System.

Day of the week	Hour of the day	VisitorID	Event	ItemID
Sun	21	189384	Transaction	299044
Mon	3	287857	Addtocart	5206
Sat	1	158090	Addtocart	10572

Compact Format

Sun	Mon	Tue	Wed	Thru	Fri	Sat	VisitorID	Event	ItemID	1	2	3	…24
1	0	0	0	0	0	0	189384	5	299044	0	0	0	…
0	1	0	0	0	0	0	287857	4	5206	0	0	1	…
0	0	0	0	1	0	0	158090	4	10572	1	0	0	…

Binary Format

Fig. 2. Compact Format to Binary Format.

We assume that user preferences are between one and five, which is the standard scale of user ratings in the literature. The value of purchasing action is set as five in this study as it represents the maximum number of prefaces a user would give to an item. The add-to-cart action is set to four as it represents lower interest than the purchase action.

B. Half-life Decay Function

Generally, Exponential decay is a mathematical term that defines the process of reducing a value with constant percentage rate over time as shown in formula (8), where n represents the decay factor as shown in equation (9) [40].

$$final\ amount = \frac{initial\ amount}{2^n} \tag{8}$$

Where

$$n = (elapsed\ time)/(half\ life) \tag{9}$$

$$half\ life = \frac{log_2(elapsed\ time)}{\log(\frac{initial\ amount}{final\ amount})} \tag{10}$$

In our scenario, we assume that the user's interest in an item decreases over time by a fixed amount of value. We use the elapsed time to determine the half-life in equation (10). The elapsed time is the period between the present date and the time when the user gave an item a rating. The half-life is defined as the number of days required to reduce the weight of a user rate by half. For rating an item, the highest rate is five and the minimum rate is one.

C. Contextual Modeling

In this phase, the filtering according to context is applied as a part of the recommendation algorithms as follows.

- Context-Aware Matrix Factorization (CAMF)

To simulate the interaction between the items and the context, Context-Aware Matrix Factorization (CAMF) adds a factor B, as illustrated in equation (11)[2]:

$$P_{u_a,i_x,c_a} = q_{i_x}^T(p_{u_a} + |N(u_a)|^{\frac{-1}{2}}) \sum_{j\in N(u_a)} y_j + B_{i_x,c_a} \tag{11}$$

These factors B are learned using the error function which can be calculated as shown in equation (12)[2]:

$$\min_{p_*,q_*,y_*,B_*} \sum_{(u_a,i_x,c_a)\in\tau} \left(r_{u_a,i_x,c_a} - q_{i_x}^T \left(P_{u_a} + |N(u_a)|^{\frac{-1}{2}} \sum_{j\in N(u_a)} y_j\right) - B_{i_x,c_a}\right)^2 + \lambda(\|q_{i_x}\|^2 + \|p_{u_a}\|^2 + \sum_{j\in N(u_a)}\|y_j\|^2 + B_{i_x,c_a}{}^2) \tag{12}$$

where N(U_a) in (11),(12) is the set of items on which the user u_a has given implicit feedback, the vector y_j in (11),(12) represents the value of implicit feedback, and *T* is the set of all items *i* that both users U_a and U_b have rated. To avoid overfitting, the regularization term. $\lambda(\|q_{i_x}\|^2 + \|p_{u_a}\|^2)$ is added to the error function.

- Context-Aware Neighborhood Based Algorithm (CA-KNN)

Similar to the Context-Unaware Neighborhood Algorithm, the Context-Aware Neighborhood Algorithm, also known as Differential Context Weighting (DCW) [42], begins by determining the similarity between users. The set *T* - the set of all items *i* that both users $u_a$and u_b have rated- is used for the summations in the similarity computations. The algorithm then maintains them along with the c1 and c2 time contexts where these ratings occurred. Equation (13) shows the incorporation of time in the similarity equation [42]:

$$\boldsymbol{sim_{ca}(u_a, u_b)} = \frac{\sum_{(i,c_1,c_2)\in T} r_{u_a,i,c_1}\, r_{u_b,i,c_2} J(c_1,c_2,\sigma_1)}{\sqrt{\sum_{(i,c_1,c_2)\in T} r^2_{u_a,i,c_1} \sum_{(i,c_1,c_2)\in T} r^2_{u_a,i,c_2} \sum_{(i,c_1,c_2)\in T} J(c_1,c_2,\sigma_1)^2}} \tag{13}$$

Where J is the weighted Jaccard metric which can be computed as given in equation (14)[42] and the set σ contains the weights for each of the potential contexts: [42]:

$$J(c_1, c_2, \sigma) = \frac{\sum_{f\in c_1\cap c_2} \sigma_f}{\sum_{f\in c_1\cap c_2} \sigma_f} \tag{14}$$

Finally, the predicted rates can be determined in the same way as the context-unaware method as illustrated in equation (15) [23]:

$$P_{u_a,i_x,c_a} = r^-_{u_a,c_a} + \frac{\sum_{(u_v,c_1)\in V}(r_{u_v,i_x,c_1} J(c_1,c_2,\sigma_2) - r^-_{u_v,c_a}) sim_{ca}(u_a,u_v)}{\sum_{(u_v,c_1)\in V}|sim_{ca}(u_a,u_v)|} \tag{15}$$

Where ($r^-_{u_v,c_a}$) is the average user ratings in the active context and the rating r_{u_v,i_x,c_1} is weighted by the Jaccard metric. the context in which the nearest neighbors have rated the item i_x stored in the set V represented by c_1.

- Context-Aware Sparse Linear Method (CA-SLIM)

Context-Aware Sparse Linear (SLIM) changes the prediction equation based on the context of the rating (c_a) applying equation (16)[41]:

$$P_{u_a,i_x,c_a} = (r^T_{u_a} + d_{c_a}) w_{i_x} \tag{16}$$

In this equation, d_{c_a} is the column in matrix D that contains the rating difference for each item-context interaction, which is referred to as the contextual rating deviations (CRD). The CRDs for all items in the context (c_a) are stored in the vector d_{c_a}By adding it to the error function, the vectors d_* and the matrix D are learned simultaneously with the matrix W, as illustrated in equation (17)[41]:

$$\min_{w_*,d_*} \sum_{(u_a,i_x,c_a)\in\tau} \left(r_{u_a,i_x} - \left(r^T_{u_a} + d_{c_a}\right) w_{i_x}\right) + \lambda_1 \|w_{i_x}\|^2 + \lambda_2 \|w_{i_x}\|_1 + \lambda_3 \|d_{c_a}\|^2 + \lambda_4 \|d_{c_a}\|_1$$
$$s.t. |w_{i_x}| \geq 0 \tag{17}$$

IV. Implementation

The data was taken from a real-world e-commerce website (Gift Shop) and is part of the "Retailrocket" public dataset [39]. A visitor can do three sorts of actions: "view," "add to cart," and "transaction.". interactions are collected over a period of 4.5 months. There are around 8k buy events and approximately 28k add-to-cart events. The day of the week and hour of the day

where the event happened were determined using the user id, item id, and Unix time. For example: "1439694000000, 1, view,100," means visitorId = 1, clicked the item with id = 100 at 1439694000000 (Unix timestamp). Then, for each of the user's events, we assume a rating value. For example, the transaction event has a value of five and the add-to-cart event has a value of four. We computed the elapsed time for each occurrence. Then we utilized it to calculate a new anticipated rate by giving less weight to the earlier event and finding the half-life value as explained in the previous section.

A. Implementation Tools

The CARSKIT Java Library is used to implement and expand the state-of-art context-aware algorithms. Numerous research [20, 45, 46] have used CARSKIT [44, 47] which is a Java-based open-source package for the context-aware recommendation. As a hardware tool, AZIZ High-Performance Computer (HPC) with 30GB RAM has been used. We modified the CARSKIT code to calculate the precision, recall, and MAP for a different number of recommended items N (from 1 to 20) to be able to track how the algorithms behave as N increases.

B. Algorithm's Hyper-parameters

The parameters applied for each of the recommendation algorithms are shown in Table II. Number of factors parameter is used in MF, If the number of latent variables (number of factors) equals one, we are selecting the most popular things with the most interactions with no regard for personalization. Increasing the number of latent variables (the number of rows or columns associated with a single product or user) would increase personalization, and hence the RS's quality. However, if the number of latent components grows too large, it will generate an overfitting problem, lowering the quality of the recommendations. To avoid the overfitting problem, we'll need some regularization terms which is used to minimize overfitting while increasing the number of latent factors. In MF and SLIM, without a predefined stopping condition, iterative steps will continue forever. A limitation on running time or the number of iterations is often used to interrupt the infinite loop. In KNN which is known as Differential Context Weighting (DCW) [42] the threshold is applied to distinguish between frequent and infrequent variables and Particle Swarm Optimization (PSO) [43] is used to determine how much weight to give to each potential context. PSO iteratively seeks to enhance potential solutions based on a specified quality metric to find the best answer to an issue [16].

Based on the default parameters for CARSKIT [44] and the suitable values for our dataset, we determined the algorithm's hyper-parameter values for our experiment as shown in Table II. We used a factor count of 10 to ensure that we had enough factors to adequately capture the variability in the data, but not too many that the training data would be overfitting. The maximum number of iterations required to obtain an accurate result is 100. Based on the "Retailrocket" dataset, which includes customer data from May 2015 to August 2015, we estimate a half-life of 43 days. The hyper-parameters for the three algorithms (MF, KNN, SLIM) with the three distinct techniques (baseline, bias, and decay) are shown in Table II.

TABLE II. EXPERIMENT CASES AND ASSUMPTIONS

	Algorithms with cases	***Hyperparameters***	***Values***
Matrix Factorization	**Case#A1: MF (Time un-aware)**	Regularization factor	default value
		# of iterations	100
		# of factors	10
	Case#A2: CAMF (Time aware (Bias))	Regularization factor	default value
		# of iterations	100
		# of factors	10
	Case#A3: Decay CAMF (Time aware Decay)	Regularization factor	default value
		# of iterations	100
		# of factors	10
		Half-life	43
Sparse Linear Method	**Case#B1: SLIM (Time un-aware)**	Regularization factor	default value
		# Of iterations	100
		# Of neighbors	10
	Case#B2: CASLIM (Time aware (Bias))	Regularization factor	default value
		# Of iterations	100
		# Of neighbors	10
	Case#B3: Decay CA-SLIM (Time aware (Decay))	Regularization factor	default value
		# Of iterations	100
		# Of neighbors	10
		Half-life	43
Nearest Neighbor Algorithm	**Case#C1: User-Based KNN (Time un-aware)**	# Of neighbors	10
	Case#C2: DCW (Time aware (Bias))	PSO parameters	default value
		Threshold	0,5
	Case#C3: Decay DCW (Time aware (Decay))	PSO parameters	default value
		Threshold	0,5
		Half-life	43

V. RESULTS

The application of the three RS algorithms Matrix Factorization (MF), K-Nearest Neighbor (KNN), and Sparse Linear Method is examined in this section (SLIM) using two different methods (Bias and Decay) as shown in Table II. The metrics used to analyze the experiment are Precision, Recall, and MAP. N is the number of suggested items, and the metrics are calculated for N=1 to N=20.

A. Case A: Matrix Factorization (MF)

This section uses two methods – Bias (traditional) and Decay- to illustrate the effect of introducing time into the matrix factorization algorithm. The accuracy, recall, and MAP for the MF are shown in Fig. 3, 4, and 5, respectively. The decay technique outperformed the Bias-CAMF and MF in terms of accuracy, recall, and MAP, with a 0.05 percent in precision,

0.45 percent recall, and 0.16 percent MAP as shown in Table III.

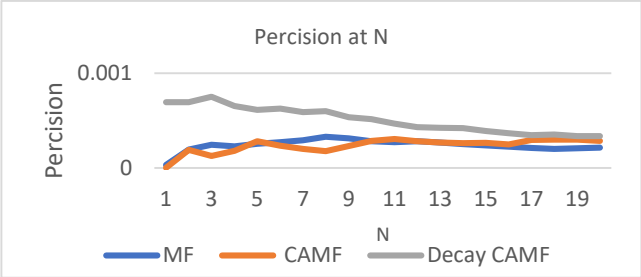

Fig. 3. MF, CAMF, and Decay CAMF Precision.

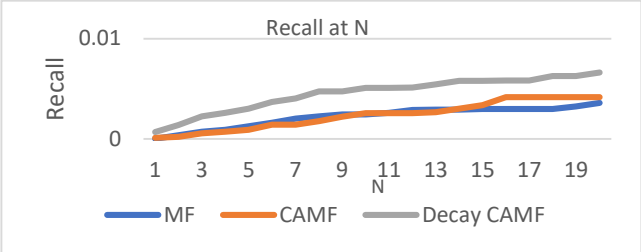

Fig. 4. MF, CAMF, and Decay CAMF Recall.

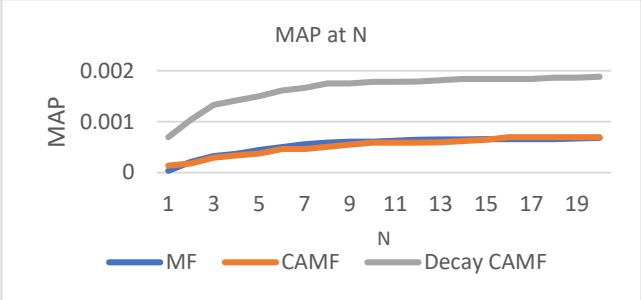

Fig. 5. MF, CAMF, and Decay CAMF MAP.

TABLE III. AVG MF PRECISION, RECALL, AND MAP

	AVG Precision percentage	AVG Recall percentage	AVG MAP percentage
MF	0.020	0.220	0.050
CAMF	0.020	0.240	0.050
Decay & CAMF	0.050	0.450	0.160

B. Case B: Sparse Lanier Method (SLIM)

Using two approaches (Bias and Decay), this section highlights the effect of adding time with the Sparse Lanier Method (SLIM) algorithm. Fig. 6, 7, and 8 depict the SLIM's precision, recall, and MAP, respectively. Table IV shows that the decay technique outperformed the time unaware SLIM and bias CA-SLIM in terms of accuracy, recall, and MAP, with precision, recall, and MAP of 0.22 percent, 1.40 percent, and 0.78 percent, respectively.

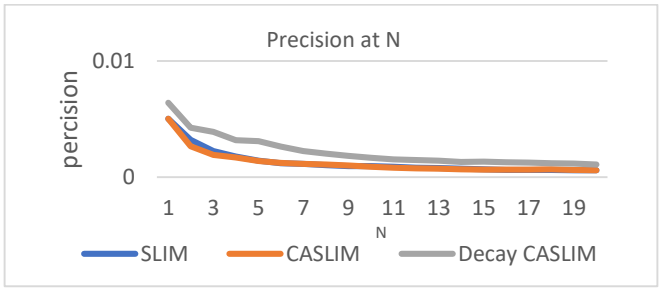

Fig. 6. SLIM, CA-SLIM, and Decay CA-SLIM Precision.

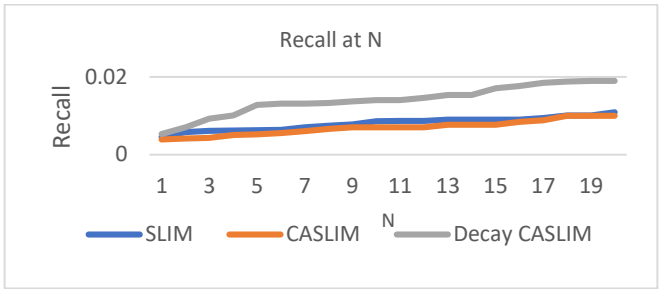

Fig. 7. SLIM, CA-SLIM, and Decay CA-SLIM Recall.

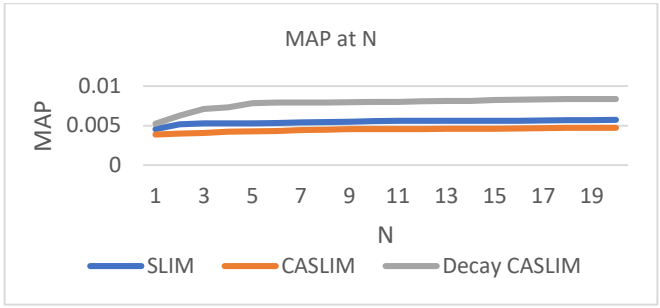

Fig. 8. SLIM, CA-SLIM, and Decay CA-SLIM MAP.

TABLE IV. AVG SLIM PRECISION, RECALL, AND MAP

	AVG Precision Percentage	AVG Recall Percentage	AVG MAP Percentage
SLIM	0.130	0.800	0.550
CA-SLIM	0.130	0.690	0.450
Decay & CA-SLIM	0.220	1.400	0.780

C. Case C: Nearest Neighbor (KNN)

Using two approaches, this section highlights the effect of adding time with the KNN algorithm. Fig. 9, 10, and 11 depict the KNN's precision, recall, and mean absolute deviation, respectively. As shown in Table V, the average results indicate that using the time context in Bias and decay method decreases the efficiency of the RS with Avg precision, recall, and MAP of 0.04%, 0.37%, and 0.10% respectively.

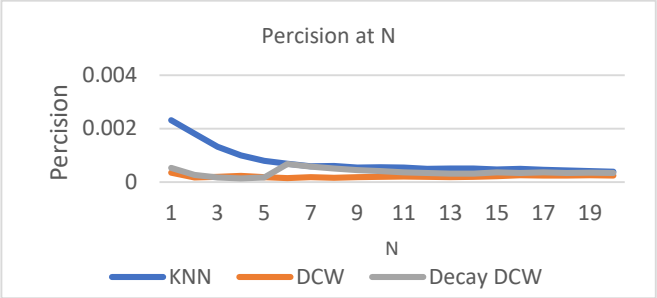

Fig. 9. KNN, DCW, Decay DCW Precision.

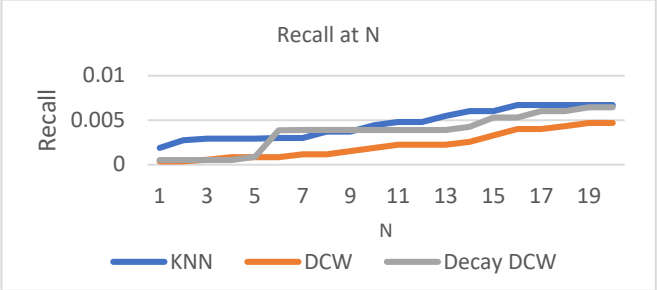

Fig. 10. Fig 1 Recall for KNN, DCW, Decay DCW

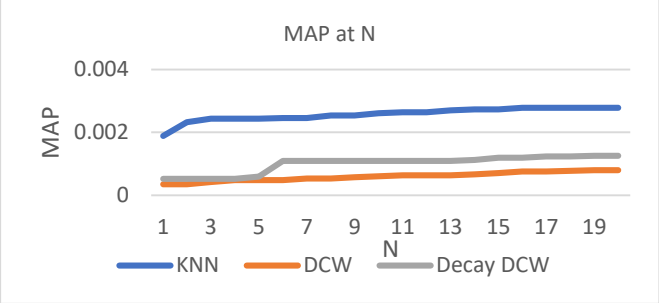

Fig. 11. Fig 2 MAP for KNN, DCW, Decay DCW

TABLE V. AVG KNN (RECALL, PRECISION, AND MAP)

	AVG Precision Percentage	**AVG Recall Percentage**	**AVG MAP Percentage**
KNN	0.070	0.460	0.260
DCW	0.020	0.220	0.060
Decay DCW	0.040	0.370	0.100

D. Half-Life Decay with All Algorithms (MF, SLIM, KNN) Comparison

The outcomes of three RS techniques (Matrix Factorization (MF), K-Nearest Neighbor (KNN), and Sparse Linear Method (SLIM)) are compared in this section: Fig. 12 and 13 illustrate the accuracy of the three RS methods with The Decay factor. With a significant difference, Decay CA-SLIM outscored the other methods. CAMF and DCW, on the other hand, have approximately similar Average-performances.

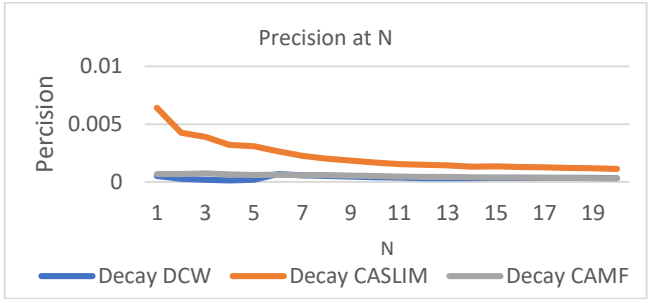

Fig. 12. The Decay Time-Aware Algorithms' Precision.

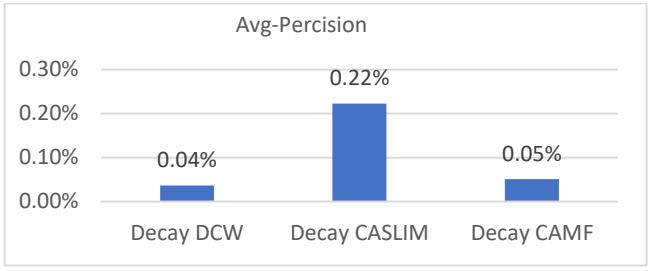

Fig. 13. The Decay Time-Aware Algorithms' Avg- Precision.

As seen in Fig. 14 and 15, Decay CA-SLIM outperformed the other algorithms by a significant margin. With 0.45 percent and 0.37 percent, respectively. while CAMF and DCW have a similar Average-performance.

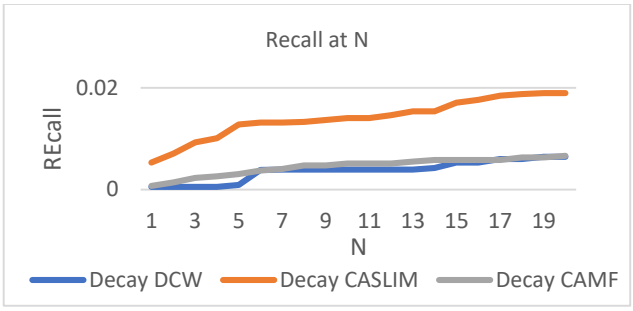

Fig. 14. The Decay Time-Aware Algorithms' Recall.

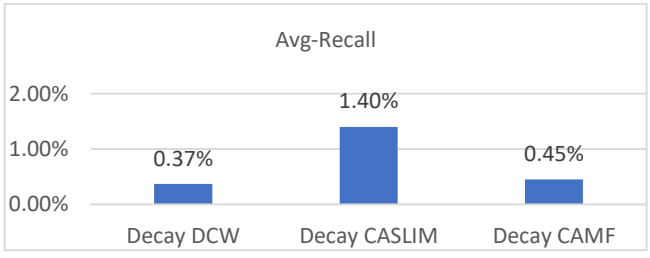

Fig. 15. The Decay Time-Aware Algorithms' Average Recall.

Fig. 16 and 17 depict the MAP for Decay DCW, Decay CAMF, and Decay CA-SLIM, respectively. For the three approaches, the avg-MAP values are 0.1 percent, 0.7 percent, and 0.16 percent, respectively.

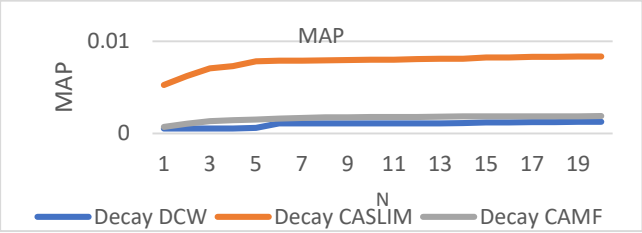

Fig. 16. The Decay Time-Aware Algorithms MAP.

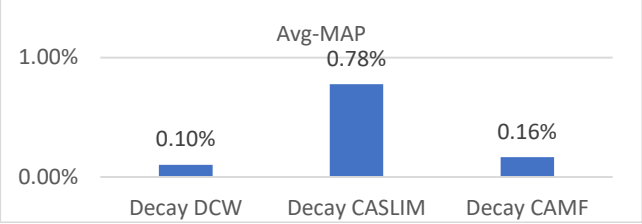

Fig. 17. The Decay Time-Aware Algorithms AVG-MAP.

E. Findings

Based on the experiment results, we conclude that in the e-commerce domain that uses purchasing data, the item-oriented, decay time-aware recommender algorithms outperform baseline context-aware algorithms and context-unaware algorithms. This conclusion is based on the fact that the overall effectiveness improves when time information is used. Furthermore, we arrive at the conclusion that the sparse linear approach, which has proven positive performance for the top-N recommendation in this research may not necessarily transition well to other datasets in the literature.

- The average findings show that the decay technique outperformed the Bias CAMF and time unaware MF, with precision, recall, and MAP of 0.05 percent, 0.45 percent, and 0.16 percent, respectively.
- The decay technique outscored the Bias CA-SLIM and time unaware SLIM approach, with precision, recall, and MAP of 0.22, 1.40, and 0.78, respectively.
- Adding the time context reduces the accuracy of the recommendation process in the KNN technique. In addition to the algorithm's incredibly high running time (weeks to execute one single run), which obstructed us from running all combinations of hyper-parameters.
- Running the KNN and SLIM with no limit on the number of neighbors had to be left out because it took weeks to execute one single run.
- Given that our utilized dataset has a density of = 0.0124 percent. The results suggest that adding decay context-awareness to KNN algorithms reduces the accuracy of the algorithms for all recommendation list lengths N. However, decay context-awareness improves the efficacy of the Sparse Linear Method (SLIM) and Matrix Factorization.
- In the online shopping area, the SLIM method outperforms both the KNN and MF algorithms using the half-life decay function. It also performs well in terms of Top-N recommendations. The increase in effectiveness was proven to be related to the average number of interactions per item[48].
- In our dataset the number of interactions per user is limited. Therefore, item-oriented algorithms like SLIM perform better than user-oriented algorithms like KNN.
- Because the SLIM method bases its predictions on item similarity coefficients, it works well on our dataset. Due to the data sparsity (limited numbers of interactions per user), some algorithms (e.g., KNN) perform poorly in terms of recommendations accuracy because it depends on the user's similarity.
- Our research results are consistent with the research [9, 49, 50] that reported increases in effectiveness when using SLIM instead of KNN or MF. this is since SLIM is specifically designed for the top-N recommendation task.

Overall, we conclude that, for the datasets with limited numbers of interactions per users the use of item-oriented decay time-aware recommender systems lead to better performance when compared to bias context aware recommender systems and time-unaware recommender systems.

VI. Conclusion

This paper used the user implicit feedback (add to cart and buy) associated with time context to improve the RS performance in online shopping systems. Bias and Decay methods were used to incorporate the time data into RSs. The experiment included incorporating time with three state-of art-algorithms (MF, KNN, and SLIM). To compare the performance of the RSs we applied three cases for each algorithm (time unaware RS, Bias time aware RS, and Decay time aware RS). Precision, Recall, and MAP were used to compare the performance between the different algorithms and between the three cases in each algorithm. As a result, Decay-MF and Decay-SLIM outperform the time Bias MF and SLIM. On the other hand, Decay-KNN reduced the accuracy of the RS compared to the context UN-aware KNN. The results that were reached from the study achieved the goal of the research, which is analyzing and finding the most appropriate algorithm in the field of e-commerce to build a time-aware recommendation system using the user's implicit feedback (purchasing data). As a future work, this study might be applied to a larger dataset with all combinations of algorithms' parameters and with unlimited number of neighbors. This study might also be expanded by combining the decay function approach with other RSs algorithms and on different domains datasets.

References

[1] Ricci, F., L. Rokach, and B. Shapira, Recommender systems: introduction and challenges, in Recommender systems handbook. 2015, Springer. p. 1-34.

[2] Ricci, F., Rokach, L. and Shapira, B, Introduction to recommender systems handbook. In Recommender systems handbook. Springer Boston, MA. , 2011: p. (pp. 1-35).

[3] Schafer, J.B., J.A. Konstan, and J. Riedl, E-commerce recommendation applications. Data mining and knowledge discovery, 2001. 5(1-2): p. 115-153.

[4] Sarwar, B., et al. Analysis of recommendation algorithms for e-commerce. in Proceedings of the 2nd ACM conference on Electronic commerce. 2000.

[5] Koren, Y., Bell, R. and Volinsky, C, Matrix factorization techniques for recommender systems. Computer, 2009: p. pp.30-37.

[6] Sarwar, B.M., et al. Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering. in Proceedings of the fifth international conference on computer and information technology. 2002.

[7] Paterek., A., Improving regularized singular value decomposition for collaborative filtering, in In Proceedings of KDD cup and workshop. 2007. p. 5–8.

[8] Paul Resnick, N.I., Mitesh Suchak, Peter Bergstrom, and John, Grouplens: an open architecture for collaborative filtering of netnews, in Proceedings of the 1994 ACM conference on Computer supported cooperative In work 1994, ACM: Riedl. p. 175–186.

[9] Karypis, X.N.a.G., Slim: Sparse linear methods for top-n recommender systems, in 2011 IEEE 11th International Conference on Data Mining. 2011, IEEE. p. 497–506.

[10] Adomavicius, G. and A. Tuzhilin, Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. IEEE Transactions on Knowledge & Data Engineering, 2005(6): p. 734-749.

[11] Burke, R., Hybrid web recommender systems, in The adaptive web. 2007, Springer. p. 377-408.

[12] Kumar, B. and N. Sharma, Approaches, issues and challenges in recommender systems: a systematic review. Indian J. Sci. Technol, 2016. 9(47): p. 1-12.

[13] Mahmood, T.a.R., Improving recommender systems with adaptive conversational strategies, in In Proceedings of the 20th ACM conference on Hypertext and hypermedia. 2009. p. (pp. 73-82).

[14] Ben-Shimon, D., et al., Recommender system from personal social networks, in Advances in Intelligent Web Mastering. 2007, Springer. p. 47-55.

[15] Dey, A.K., Understanding and using context. Personal and ubiquitous computing, 2001. 5(1): p. 4-7.

[16] Schmidt, A., M. Beigl, and H.-W. Gellersen, There is more to context than location. Computers & Graphics, 1999. 23(6): p. 893-901.

[17] Adomavicius, G. and A. Tuzhilin, Context-aware recommender systems, in Recommender systems handbook. 2011, Springer. p. 217-253.

[18] Campos, P.G., F. Díez, and I. Cantador, Time-aware recommender systems: a comprehensive survey and analysis of existing evaluation protocols. User Modeling and User-Adapted Interaction, 2014. 24(1-2): p. 67-119.

[19] De Borba, E.J., I. Gasparini, and D. Lichtnow. Time-aware recommender systems: a systematic mapping. in International Conference on Human-Computer Interaction. 2017. Springer.

[20] van Kortenhof, B.L., Context-Aware Recommender Systems in the E-commerce Domain. 2017.

[21] Introduction to Latent Matrix Factorization Recommender Systems Available from: towardsdatascience.com.

[22] Guibing Guo; Jie Zhang; Zhu Sun; and Neil Yorke-Smith. In Posters, D., Librec: A java library for recommender systems., in Late-breaking Results and Workshop Proceedings of the 23rd International Conference on User Modeling, Adaptation and Personalization. 2015.

[23] Sánchez-Moreno, D., Y. Zheng, and M.N. Moreno-García. Incorporating time dynamics and implicit feedback into music recommender systems. in 2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI). 2018. IEEE.

[24] de Zwart, T., Time-Aware Neighbourhood-Based Collaborative Filtering. 2018.

[25] de Borba, E.J., I. Gasparini, and D. Lichtnow. The Use of Time Dimension in Recommender Systems for Learning. in ICEIS (2). 2017.

[26] Luo, J., et al., A context-aware personalized resource recommendation for pervasive learning. Cluster Computing, 2010. 13(2): p. 213-239.

[27] Karahodza, B., H. Supic, and D. Donko. An Approach to design of time-aware recommender system based on changes in group user's preferences. in 2014 X International Symposium on Telecommunications (BIHTEL). 2014. IEEE.

[28] Gallego, D., et al. A model for generating proactive context-aware recommendations in e-learning systems. in 2012 Frontiers in Education Conference Proceedings. 2012. IEEE.

[29] Inuzuka, K., T. Hayashi, and T. Takagi. Recommendation system based on prediction of user preference changes. in 2016 IEEE/WIC/ACM International Conference on Web Intelligence (WI). 2016. IEEE.

[30] Rezaeimehr, F., et al., TCARS: Time-and community-aware recommendation system. Future Generation Computer Systems, 2018. 78: p. 419-429.

[31] Feng, H., et al., Personalized recommendations based on time-weighted overlapping community detection. Information & Management, 2015. 52(7): p. 789-800.

[32] Gueye, M., T. Abdessalem, and H. Naacke, Dynamic recommender system: using cluster-based biases to improve the accuracy of the predictions, in Advances in Knowledge Discovery and Management. 2016, Springer. p. 79-104.

[33] Luo, C., X. Cai, and N. Chowdhury. Self-training temporal dynamic collaborative filtering. in Pacific-Asia Conference on Knowledge Discovery and Data Mining. 2014. Springer.

[34] Chen, J., et al., A Temporal Recommendation Mechanism Based on Signed Network of User Interest Changes. IEEE Systems Journal, 2019.

[35] Arora, R.M.A.T.A., Temporal Recommendations for Discovering Author Interests, in 2019 Twelfth International Conference on Contemporary Computing (IC3) 2019: Noida, India. p. pp. 1-6.

[36] Benlamri, R. and X. Zhang, Context-aware recommender for mobile learners. Human-centric Computing and Information Sciences, 2014. 4(1): p. 12.

[37] Chen, W., et al., A hybrid recommendation algorithm adapted in e-learning environments. World Wide Web, 2014. 17(2): p. 271-284.

[38] Wei, S., N. Ye, and Q. Zhang. Time-aware collaborative filtering for recommender systems. in Chinese Conference on Pattern Recognition. 2012. Springer.

[39] Retailrocket recommender system dataset. Available from:https://www.kaggle.com/retailrocket/ecommerce-dataset.

[40] Ludwig–Maximilians, Demonstration of the exponential decay law using beer froth. EUROPEAN JOURNAL OF PHYSICS, 2001: p. 21-26.

[41] Yong Zheng, B.M., and Robin Burke., Cslim: Contextual slim recommendation algorithms, in Proceedings of the 8th ACM Conference on Recommender Systems. 2014., ACM. p. 301–304.

[42] Yong Zheng, R.B., and Bamshad Mobasher, Recommendation with differential context weighting, in In International Conference on User Modeling, Adaptation, and Personalization. 2013, Springer. p. pages 152–164.

[43] Russell, K.J.a.E., Particle swarm optimization, in 1995 IEEE International Conference on Neural Networks,. 1995. p. 1942–1948.

[44] Yong Zheng, B.M., and Robin Burke. , Carskit: A java-based context-aware recommendation engine. . IEEE International Conference on Data Mining Workshop (ICDMW), 2015: p. 1668–1671.

[45] Al Jawarneh, I.M., Bellavista, P., Corradi, A., Foschini, L., Montanari, R., Berrocal, J. and Murillo, J.M., A Pre-Filtering Approach for Incorporating Contextual Information Into Deep Learning Based Recommender Systems. IEEE Computer Society, 2020: p. 40485-40498.

[46] Ilarri, S., Trillo-Lado, R. and Hermoso, R., Datasets for context-aware recommender systems: Current context and possible directions, in 2018 IEEE 34th International Conference on Data Engineering Workshops (ICDEW). 2018,, IEEE. p. 25-28.

[47] Zheng, Y., A User's Guide to CARSKit. 2015, arXiv preprint

[48] Pedro G. Campos Soto. , , . Temporal models in recommender systems: An exploratory study on different evaluation dimensions. 2011, Universidad Autónoma de Madrid: Madrid.

[49] Hoslim, E.C.a.G.K., Higher-order sparse linear method for top-n recommender systems., in Pacific-Asia Conference on Knowledge Discovery and Data Mining. 2014, Springer. p. 38–49.

[50] Karypis., E.C.a.G., Local item-item models for top-n recommendation, in Proceedings of the 10th ACM Conference on Recommender Systems. 2016, ACM. p. 67–74.

[51] A. Hassan, E. Fadel, and N. Akkari, Time-Aware Recommender System For E-Commerce Applications, 2020. p. 534-542.

[52] Wang, D., Xu, D., Yu, D., & Xu, G. (2021). Time-aware sequence model for next-item recommendation. Applied Intelligence, 51(2), 906-920.

[53] Yang, D., Nie, Z. T., & Yang, F. (2021). Time-aware CF and temporal association rule-based personalized hybrid recommender system. Journal of Organizational and End User Computing (JOEUC), 33(3), 19-34.